Instructions how to use the DAQ in a MINIBALL experiment

R. Lutter

September 12, 2013

Abstract

This document describes how to use the MARaBOOU data acquisition system in a MINIBALL experiment.

Contents

1 Getting started

1.1 Login to the DAQ computer

To login into the DAQ computer do:

At CERN: `ssh miniball@pcepuis20.cern.ch`
or at Cologne: `ssh miniball@minidaq.ikp.uni-koeln.de`

Make sure that your working directory is the one prepared for the current experiment.
A `pwd` command should give something like `/d1/miniball/<my_working_directory>`:

`/d1/miniball/cern-040719` for example

Use `cd /d1/miniball/<my_working_directory>` in case you are in the wrong place.
To start a new session in a new working directory refer to 1.12 .

From now on it is assumed that you are logged into the DAQ computer.

1.2 How to set up and control a list-mode run

To learn how to generate your code and to compile analysis and readout parts, respectively, see 1.10.

To start the control GUI type:

`C_analyze`

Once the GUI has popped up (fig. 1) you should check if all settings are as expected:

- Set `RUN` number appropriate. It will be incremented after each run.
- Choose `TcpIp` to connect to the PowerPC and the `VME` crate.
- Choose `ppc-0` from the `Master` list.
  This will set the `Readout` processor to `ppc-0` automatically.
- Set `Directory` to `<my_working_directory>/ppc`
- The name of the `ROOT` file to store histograms should be `histsRUN.root`,
  `RUN` will on start be substituted by the current run number.
- Set `Mapped name` to `none`
- Enable or disable raw data output. The name of the output file should be `runRUN.med`,
  extension `.med` is mandatory to produce med formatted data.

If you made some changes to these settings you should save them pressing (fig. 2)

`Save Setup` → `Save current settings`

Now press

`Clear MBS`

This should stop all pending MBS processes and put MBS into an idle state. You should end up with the message

```
c_ana:  Ok, all MBS processes disappeared
```

In case of problems you have to reset MBS manually (see 1.9).

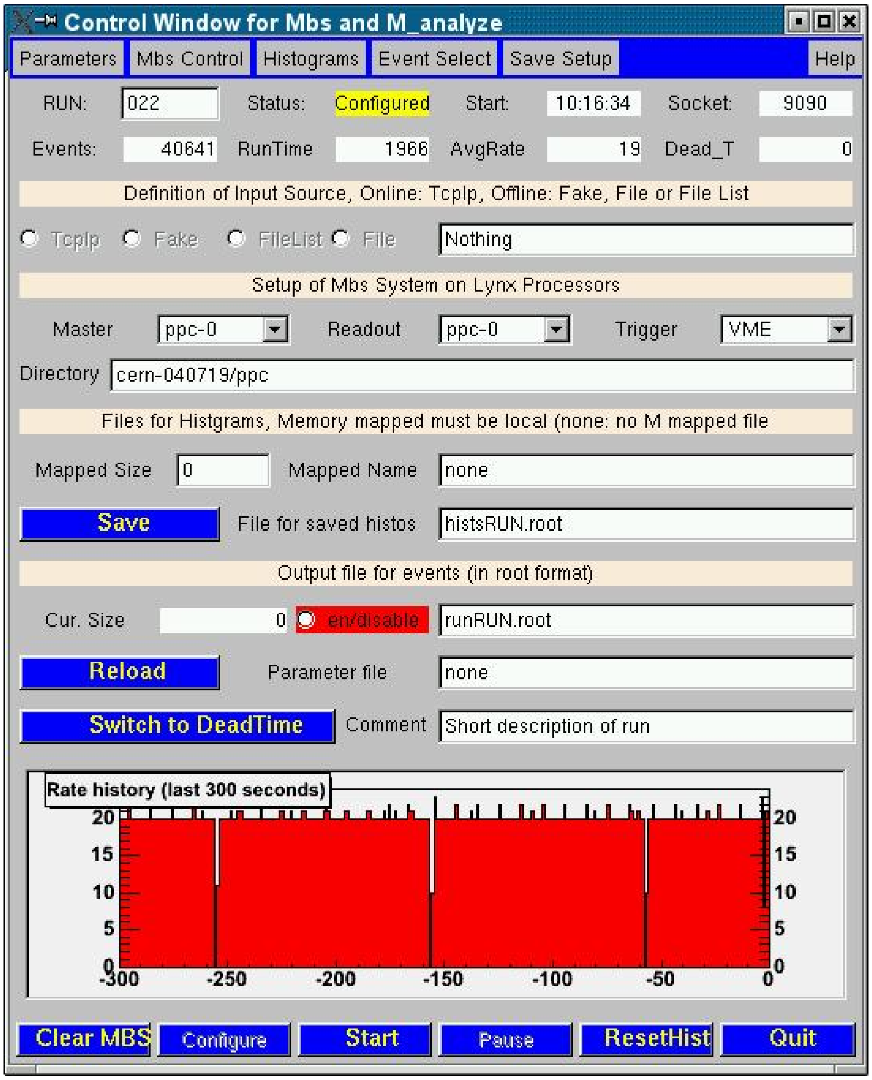


Figure 1: C_analyze - GUI to control a list-mode run

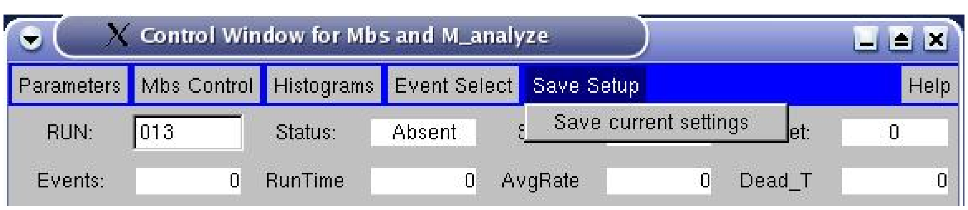


Figure 2: C_analyze: how to save setup

To configure `MBS` for your experiment press

`Configure`

After having selected wether a file should be written or not press

`Start`

To stop the run press

`Stop`

This will close the list-mode data file and write all histograms to a `ROOT` file.

1.3 How to run in AutoFile mode

As file sizes are restricted to `2 GB` one has to keep files at sizes below this limit.
You may activate the `AutoFile` mode to split the raw data stream into several files in a production run. Choose from the menu bar (fig. 3)

`Parameters` → `Maximum output file size`

insert the value you want in MB, then activate

`Parameters` → `Enable automatic restart after max file size`

This will cause `C_analyze` to stop the run as soon as the given file size is reached and to continue with the next run (run number will be increased).
As the size check is done every second you should set the maximum file size a true bit below the limit of `2 GB` (let's say to `1800`) to give `C_analyze` a chance to stop before the limit is reached. Otherwise the resulting file may be truncated.

To leave `AutoFile` mode simply press `Stop`.

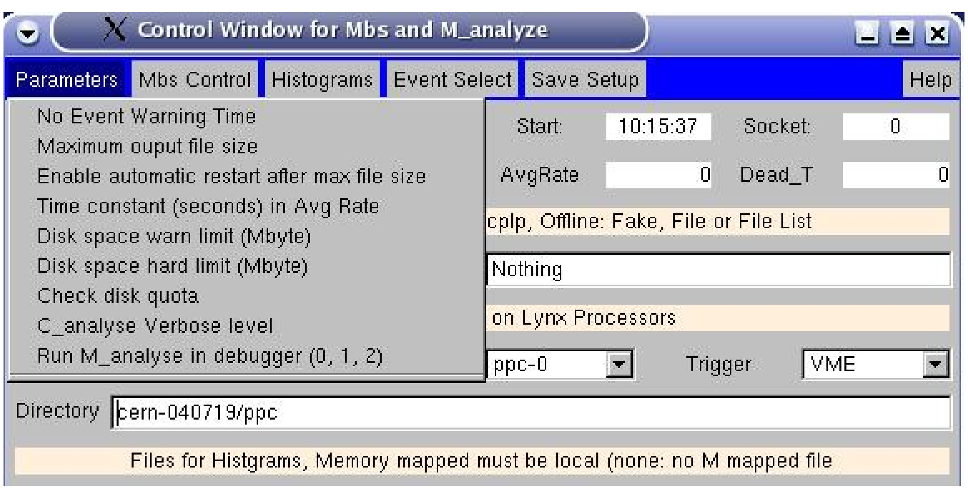


Figure 3: `C_analyze`: how to set autoFile mode

1.4 Display of scaler data

To display contents of the `VME` scalers as well as the internal dgf scalers open two separate `xterm` windows. Then type

`scaler.sh` (**without** preceeding "./"!)

to display the `VME` scalers, and

`dgfscaler.sh` (**without** preceeding "./"!)

to display dgf scalers, respectively.
Scaler data on the screen will be updated every second.

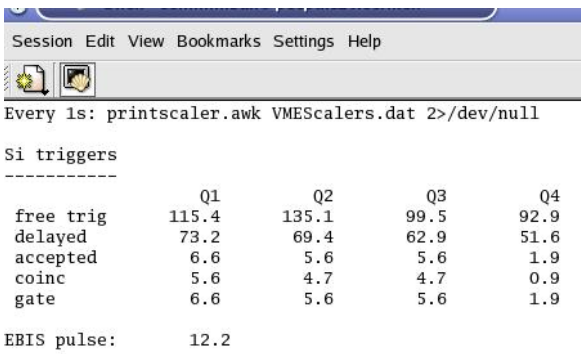


Figure 4: `Display of scaler data`

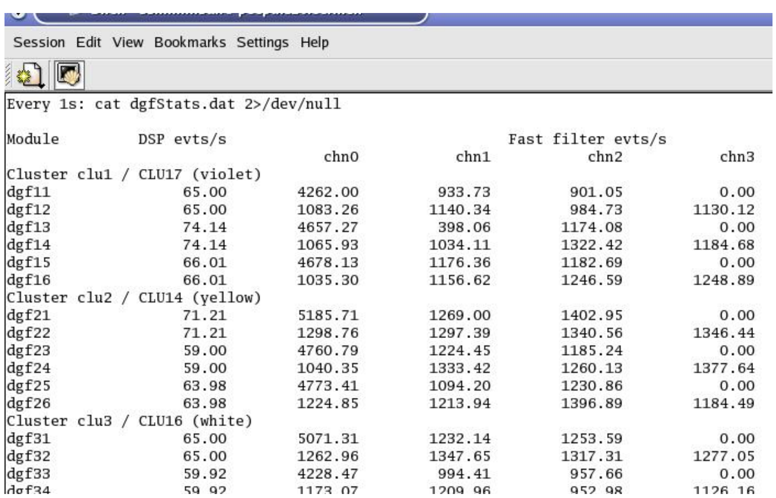


Figure 5: `Display of internal dgf scalers`

1.5 PPAC beam monitor

To show the PPAC profile simply type

`ppac.C`

This will display PPAC currents for X and Y strips, respectively, with a repetition rate of 1 per second.

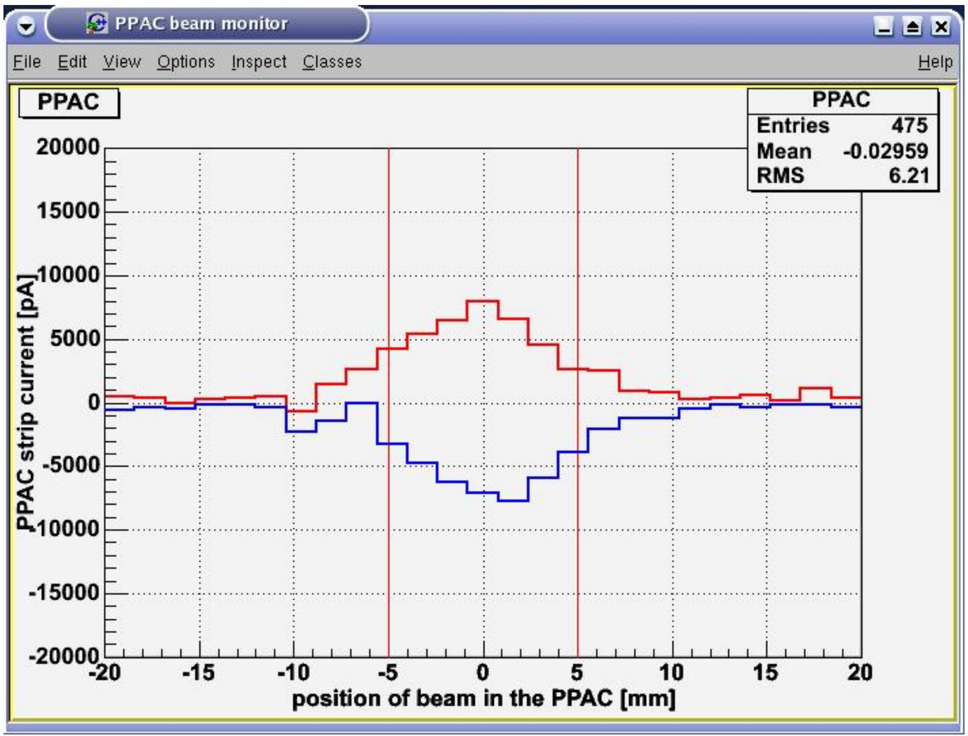


Figure 6: `PPAC beam monitor`

1.6 Beam rate monitor

To start the rate monitor type

`rateMon.C`

It displays counting rates for DGF cores as well as the beam dump detector. An alarm may be triggered if the rate goes below a given threshold.

Counting rates are taken from a file produced by function `TUsrEvtReadout::PeakCheck()` in the online daq process (see code in file `udef/BuildEvent.cxx`). It integrates data in two windows given by definitions in `.rootrc`:

```
TMrbAnalyze.PeakCheck.eMin: 276
TMrbAnalyze.PeakCheck.eMax: 282
TMrbAnalyze.PeakCheck.ratioFact: 1
TMrbAnalyze.PeakCheck.eMin2: 639
TMrbAnalyze.PeakCheck.eMax2: 632
TMrbAnalyze.PeakCheck.ratioFact2: 1
```

Therefore one has to set these values properly before starting the daq process.

`rateMon.C` provides the following commands:

`start(range, avgShort, avgLong [, withBeamDump])`
start rate display for DGF cores (and beam dump detector)

`range`	history/histogram range (s)
`avgShort`	average time (s) - short term
`avgLong`	average time (s) - long term
`withBeamDump`	show beam dump rates if `kTRUE`

`stop()` stop display

`cont()` continue display

`startwd(thresh, avgTime)`
start watchdog to trigger alarm if beam below threshold

`thresh`	trigger threshold for "beam low" alarm
`avgTime`	average time (s)

`stopwd()` stop watchdog

`bye()` exit program

Any of these commands may be given after the ROOT prompt manually. To start automatically with predefined settings you may create a startup file named `.rateMon.rc` in your working directory:

```
{
        start(100, 17, 84, kFALSE);
        startwd(1000, 19);
}
```

(Keep in mind: ROOT commands have to be enclosed in curly braces `{...}`!)

In this example `rateMon.C` will automatically start the rate display:

- history range is `100`

- averaging will be done over `17` and `84` seconds, respectively
- as there is no beam dump detector in the experimental setup only core rates will be displayed
- if core rates go below a threshold of `1000` averaged over `19` seconds an alarm will be issued

1.7 Laser on/off monitor

To show the laser on/off scaler data type

`laser.C`

This will give you a plot of the laser data over the last 20 minutes with a binning of 4 seconds.

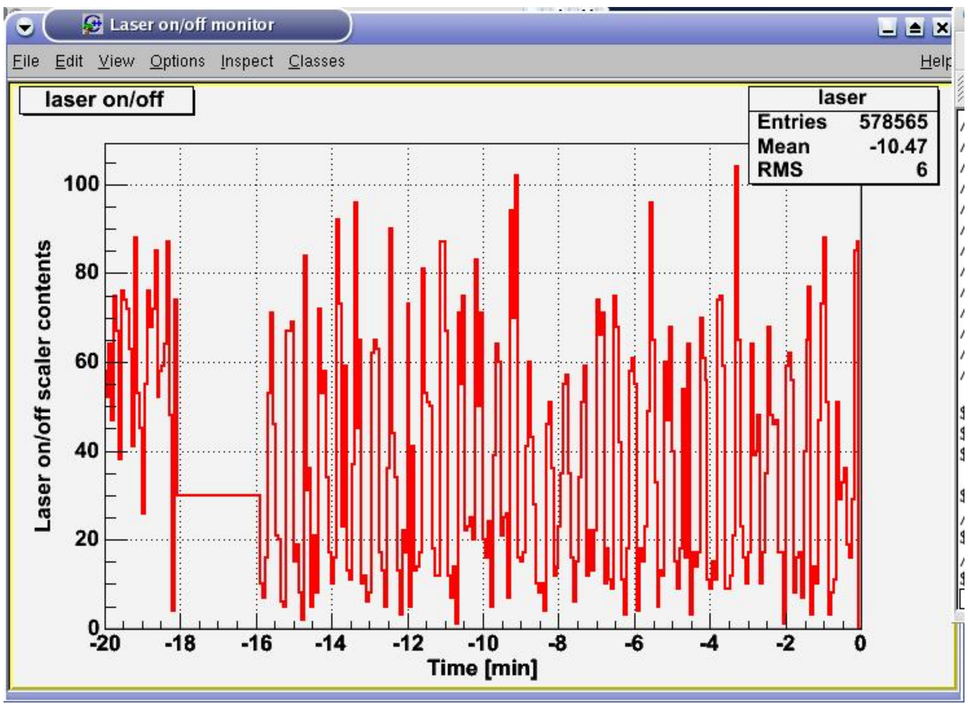


Figure 7: `Laser on/off monitor`

1.8 Display of histograms

To look at spectra you have to start the Histogram Presenter:

`HistPresent`

To connect to a running `C_analyze` click on `Hists from M_analyze`.

host should be `localhost`,
port has to be `9090`.

Be aware that only **online** histograms may be accessed this way, only as long as data acquisition is running. To look at histograms saved from previous runs click on

`Show Filelist`.

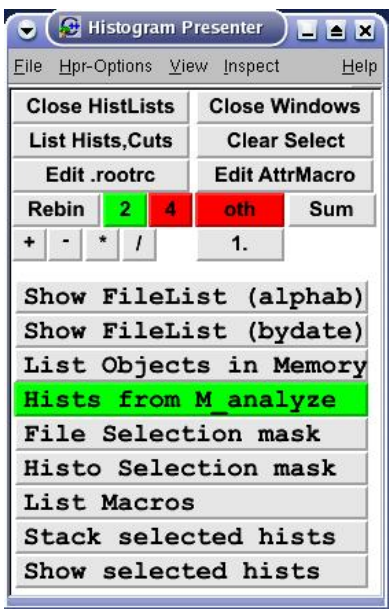


Figure 8: `HistPresent`

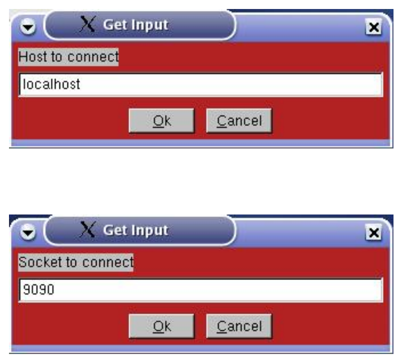


Figure 9: Connect to `M_analyze`

1.9 How to reset MBS manually

In case the `Clear MBS` button of `C_analyze` doesn't work as expected you have to reset `MBS` manually. (Open a new `xterm`, then) type

`rsh ppc-0` to login to the ppc.

Then change directory to the current experiment:

cd <my_working_directory>/ppc (example: `cd cern-040719/ppc`)

Then type

`resmbs`

This should kill all MBS processes (the ones starting with `m_` in the name when doing `ps ax`).

If there is some error message during `resmbs` ("device busy" or similar) you should do a `ps ax` and look for the line containing the `m_prompt` process:

```
53 1 53 18 168 8 68 0.05 miniball W /mbs/deve/bin_RIO2/m_prompt
```

Pick the first number from this line then and kill this process by typing

`kill <proc_no>`   (this example: `kill 53`)

Type `logout` to leave the ppc session.

1.10 How to generate and to compile code

To update the DGF cluster settings you have to edit files

`cluster.def`	settings for **active** clusters
`cluster-void.def`	settings for clusters which are currently **inactive** but have DGF modules assigned
`other-dgfs.def`	settings for other DGF modules such as time stamper, beam dump detector, etc.

The file format is adopted from Nigel Warr's Miniball Configuration sheet. See 5.3.1 for a description.

CAVEAT: Make sure that **all existing** DGF modules are defined in these files. Otherwise any uninitialized DGF will spoil its CAMAC crate!

To generate your code from the config file simply type

`./Config.C`

This will generate all code files needed for the experiment (fig. 10). Existing files will be overwritten.

To compile the `ROOT` part of the code (running on your desktop under `Linux`) type

`make -f DgfAnalyze.mk clean all`

This will compile and link program `M_analyze` which is then used by the control GUI `C_analyze`.
This step has to be repeated whenever you made changes in the configuration or in the user part of your code (code files residing in the `udef` subdirectory).

To compile the readout part of your code (running under `MBS`) call `C_analyze`, then click on

`Mbs Control` → `Compile readout function` (fig. 11)

Alternatively you may compile the readout code in the ppc directly:

```
rsh ppc-0
cd <my_working_directory>/ppc
```

```
make -f DgfReadout.mk clean all
logout
```

This will produce MBS task m_read_meb in subdirectory ppc.
Repeat this step whenever the hardware config has changed (e.g. number and position of VME and CAMAC modules).

```
[Loading MARaBOU's config libs from /usr/local/marabou_new/lib]
[Loading DGF libs from /usr/local/marabou_new/lib]
TMrbLogger::Open(): Writing (error) messages to log file marabou.log
Config.C: Configuring for ONLINE data acquisition
TMrbConfig::MakeDefined(): [__EVENT_BUILDING_ON__] Start event building : FALSE
TMrbConfig::MakeDefined(): [__CHECK_CONDITIONS__] Check window conditions : FALSE
TMrbConfig::MakeDefined(): [__WITH_CDE_DETECTOR__] CDE detector used : FALSE
TMrbConfig::MakeDefined(): [__WITH_PATTERN_UNIT__] Pattern unit used : TRUE
TMrbConfig::MakeDefined(): [__WITH_BEAMDUMP_DETECTOR__] Beamdump detector used : TRUE
TMrbConfig::MakeDefined(): [__WITH_CPTM_MODULE__] CPTM module used : FALSE

DGF Cluster Data:
  # id   hex    V color      AF side   height  angle     C N1 N2 DGFs
--------------------------------------------------------------------------------
  1 17 A  625 4500 violet     C4 right  down    backward  1  3  4 dgf11 dgf12
  1 17 B  621 3500 violet     C4 right  down    backward  1  5  6 dgf13 dgf14
  1 17 C  614 4000 violet     C4 right  down    backward  1  7  8 dgf15 dgf16

  3 16 A  620 3500 white      C2 right  down    forward   1 17 18 dgf31 dgf32
  3 16 B  619 4000 white      C2 right  down    forward   1 19 20 dgf33 dgf34
  3 16 C  632 4000 white      C2 right  down    forward   1 21 22 dgf35 dgf36

  6 22 A  611 3500 brown      A4 left   down    forward   2 17 18 dgf61 dgf62
  6 22 B  613 3500 brown      A4 left   down    forward   2 19 20 dgf63 dgf64
  6 22 C  618 3500 brown      A4 left   down    forward   2 21 22 dgf65 dgf66

  7 14 A  628 4000 blue       A2 left   down    backward  3 11 12 dgf71 dgf72
  7 14 B  601 4000 blue       A2 left   down    backward  3 13 14 dgf73 dgf74
  7 14 C  629 4000 blue       A2 left   down    backward  3 15 16 dgf75 dgf76

  2  0 A    0    0 void       xx void   void    void      1 11 12 dgf21 dgf22
  2  0 B    0    0 void       xx void   void    void      1 13 14 dgf23 dgf24
  2  0 C    0    0 void       xx void   void    void      1 15 16 dgf25 dgf26

  4  0 A    0    0 void       xx void   void    void      2  4  5 dgf41 dgf42
  4  0 B    0    0 void       xx void   void    void      2  6  7 dgf43 dgf44
  4  0 C    0    0 void       xx void   void    void      2  8  9 dgf45 dgf46

  5  0 A    0    0 void       xx void   void    void      2 11 12 dgf51 dgf52
  5  0 B    0    0 void       xx void   void    void      2 13 14 dgf53 dgf54
  5  0 C    0    0 void       xx void   void    void      2 15 16 dgf55 dgf56

  8  0 A    0    0 void       xx void   void    void      3 17 18 dgf81 dgf82
  8  0 B    0    0 void       xx void   void    void      3 19 20 dgf83 dgf84
  8  0 C    0    0 void       xx void   void    void      3 21 22 dgf85 dgf86

TMrbXia_DGF_4C: Generating code for XIA Release v2.7mb (May-2002) (binary)
  9  0 A    0    0 ts        xx void   void    void      3  4  0 dgf91        // time stamping dgfs
 10  0 A    0    0 bd        xx void   void    void      3  9  0 dgf101       // beam dump
TMrbConfig::CheckConfig(): Check done - no inconsistencies encountered
[DgfReadout.c: C code (VME/CAMAC readout) for MBS]
[DgfReadout.h: C definitions for MBS]
[DgfReadout.mk: Makefile (LynxOs)]
[DgfAnalyze.cxx: C++ code to be used with ROOT]
[DgfAnalyze.h: C++ class definitions]
[DgfAnalyzeLinkDef.h: CINT directives]
[DgfAnalyze.mk: Makefile (ROOT)]
[DgfAnalyzeGlobals.h: User's global pointers (ROOT)]
[DgfLoadLibs.C: Macro to load libs for an interactive ROOT session]
[DgfCommonIndices.h: Common indices for analysis AND readout]
[DgfAnalyze.html: HTML documentation, class index]
[.DGFControl.rc: Environment settings]
[.mbssetup: Definitions to perform MBS setup]
TMbsSetup::Open(): Creating file .mbssetup
TMbsSetup::CopyDefaults(): Copied 9 resource(s) matching "TMbsSetup.EvtBuilder.*"
TMbsSetup::CopyDefaults(): Copied 23 resource(s) matching "TMbsSetup.Readout1.*"
TMbsSetup::Save(): Resource data saved to file .mbssetup
[No errors during config step]
```

Figure 10: Config.C: generating code files

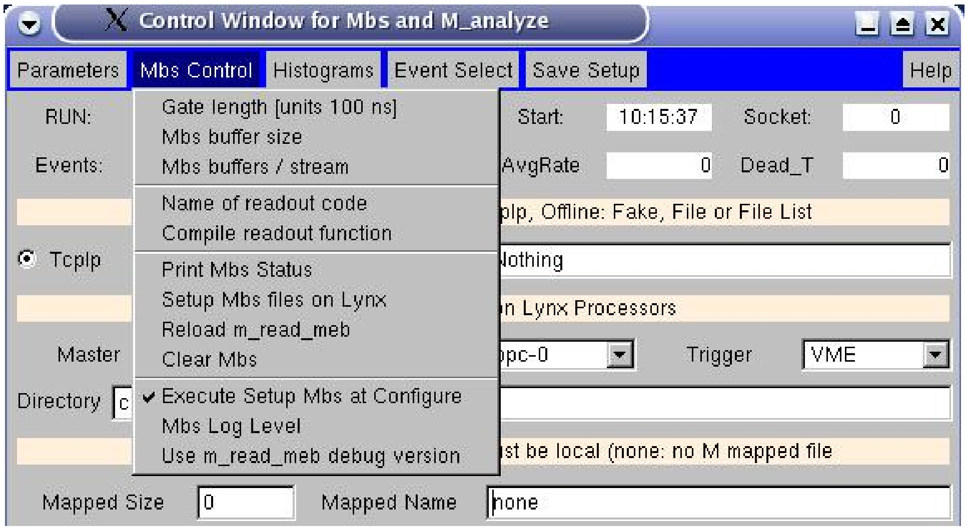


Figure 11: `C_analyze`: how to compile readout task

1.11 How to establish a directory for an offline session

In an online run only a few diagnostic tests may be performed beside data taking. To evaluate data one has to establish an offline session in parallel.

To start an offline session you first have to create directories and subdirectories and to copy and link files which are identically used in online and offline sessions. script `mkoffl` will do the job:

```
cd /d1/miniball
mkoffl <online_dir>
```

It creates an offline directory `<online_dir>-offline`. This naming convention will later on be used by script `Config.C` to distinguish between online and offline.

`mkoffl` will do the following:

- create subdirectories `<online_dir>-offline/udef` and `<online_dir>-offline/config`
- copy `<online_dir>/.rootrc`
- copy contents of subdir `<online_dir>/config`
- copy "ifdefs" in `<online_dir>/SetCppIfdefs.C`
- copy calibration files `<online_dir>/*.cal`
- link config file `<online_dir>/Config.C`

Now one has to create/modify files to meet offline requirements:

- modify entries in `SetCppIfdefs.C` (enable event building and window check for example)
- book additional histograms in file `BookHistogramsOffline.C`
- define window settings in `DefineVarsAndWdws.C`
- place your analysis code in subdirectory `udef`: `udef/Analyze.cxx + .h`

You should then be able to perform the config step and to compile and link your code (see 1.10):

```
./Config.C
make -f DgfAnalyze.mk clean all
```

Now start `C_analyze` and attach to the `.med` file which is being actually produced in the online directory. You may thus perform a "quasi-online" run in parallel to the real online data acquisition.

1.12 How to start a new session in a new directory

To start a new experiment in a new working directory go one level up:

```
cd /d1/miniball
```

Then type

```
mknew <old_dir> <new_dir>
```

where `<old_dir>` is the directory you worked before, `<new_dir>` is the one you want to start a new experiment in.

`mknew` will do the following:

- copy `<old_dir>/*.C` to `<new_dir>`
- copy `<old_dir>/.rootrc` to `<new_dir>`
- copy `<old_dir>/*.def` to `<new_dir>`
- copy subdirectories `<old_dir>/udef` and `<old_dir>/config` to `<new_dir>`
- create subdirectory `<new_dir>/ppc`
- perform a config step calling `<new_dir>/Config.C` (2x `:-(`)
- compile and link program `M_analyze` to be used by `C_analyze`

You should then be able to run `C_analyze`. Follow instructions in 1.2 to setup your experiment properly. Don't forget to re-compile the `MBS` readout task before starting data acquisition (1.10).

1.13 How to produce an ascii dump of .med data

There is a tool called `mbs2asc` which may be used to dump `.med` data to ascii for debugging purposes.

```
Usage: mbs2asc [-r <rcFile>] [-n <maxEvents>] [-t <rdoTrig>] [-f <dgfFmt>]
                                                [-d <sevtType>] [-v] <mbsFile>

mbsFile        raw data file (extension .lmd or .med)

-r <rcFile>    use indices and defs from <rcFile> (default: no defs)
-n <maxEvents> process <maxEvents> only (default: end of file)
-t <rdoTrig>   readout trigger is <trigger> (default: 1)
               (there may be more than one option "-t" in case of multiple triggers)
-f <dgfFmt>    use DGF-4C format descriptor <dgfFmt> in case of format errors
-d <sevtType>  raw data file contains subevent dumps rather than original mbs data (extension .dmp)
               <sevtType> = "dgf" or "caen" (default: none)
-v             turn on verbose mode: output hex dump in addition to other data
```

For example, command

```
mbs2asc -r .DGFControl.rc -n 10 -v run140.med | less
```

will produce output

```
# Program                 : mbs2asc
# Arguments               : -r .DGFControl.rc -n 10 -v run140.med
# Input                   : run140.med
# Indices & defs          : .DGFControl.rc
# Event trigger(s)        : 1
# Max number of events    : 10
# Verbose mode            : on
 ...
MBS EVT    10     1   14  1 1594049            # start acquisition (trigger #14)
MBS EVT    10     1    1  2 1594050            # readout (trigger #1)
MBS SEVT 9000     1  999                       # subevent "Time stamp"
MBS SEVT   10    23    2                       # subevent "clu2"
DGF BUF   36      7  257  0       0   65167   65167 # 0024 0007 1101 0000 0000 fe8f # module "dgf21"
DGF EVT    7                      1   17443   82979 # 0007 0001 4423
DGF CHN    0   2663                   17468   83004 # 0008 443c 0a67 1618 1af0 0000 0000 0000
DGF CHN    1      0                   17468   83004 # 0008 443c 0000 0000 0000 ffd5 000b 0000
DGF CHN    2      0                   17468   83004 # 0008 443c 0000 0daf 0000 002c 0003 0000
DGF BUF   45      8  257  0       0   65167   65167 # 002d 0008 1101 0000 0000 fe8f # module "dgf22"
DGF EVT   15                      1   17443   82979 # 000f 0001 4423
DGF CHN    0   2541                   17468   83004 # 0008 443c 09ed 0000 0000 ff70 0021 0000
DGF CHN    1      0                   17468   83004 # 0008 443c 0000 0c80 0000 ffe8 0006 0000
DGF CHN    2      0                   17468   83004 # 0008 443c 0000 0000 0000 0015 001d 0000
DGF CHN    3      0                   17468   83004 # 0008 443c 0000 0000 0000 ffee 0018 0000
DGF BUF   36      9  257  0   0       65166   65166 # 0024 0009 1101 0000 0000 fe8e # module "dgf23"
DGF EVT    7                  3        5923  202531 # 0007 0003 1723
DGF CHN    0  12127                    5947  202555 # 0008 173b 2f5f 1770 1e57 0000 0000 0000
DGF CHN    1      0                    5947  202555 # 0008 173b 0000 0000 0000 ffe1 0010 0000
DGF CHN    2      0                    5947  202555 # 0008 173b 0000 0000 0000 ffd7 0016 0000
```

2 Set up and control XIA DGF-4C modules

`DGFControl` is a program to set up the DGFmodules for the DAQ. It is NOT necessary to restore the `DGF` parameter settings for each run. Only if the `CAMAC` crates have been switched off or the `DGF` modules have been booted they have lost their parameters. Fortunately the settings have been saved and can be restored from file (don't forget to save your settings after a change!).

CAVEAT: Connecting to `DGF` modules via `DGFControl` may disturb a running data acquisition. Be sure that no daq is running or that you pressed `Stop` or `Pause` in the `C_analyze` GUI to stop it.

Open a new `xterm` window. Then type

`DGFControl`

The main (`system`) tab should then show up at your screen (fig. 12.

press `Restart ESONE` to (re-)start the `CAMAC` server

press `Reload DGFs` to download the volatile DSP and FPGA code

(this has to be done whenever the `CAMAC` crates has been switched off)

press `Connect`

Then open the `Restore` tab and reload the appropriate parameter settings.

Visit shortly the `Files` and `Modules` tabs to check if the right DSP/FPGA code has been downloaded and the DSP parameters are correct. If the file names differ from what you expect you'll have to set the proper values in your `.rootrc` file and to start over. If the shaping times for the DGFs are not 6.8 us peaking and 2 us gap time, you probably forgot to restore parameters.

The list below describes what the different tabs in `DGFControl` are meant for.

- `System` (fig. 12)
  Restart `ESONE/CAMAC` server, reload (= boot) dgf modules, connect to modules if server is still running
- `Modules` (fig. 13)
  Control and change modules settings, one sheet per module/channel
- `Params`
  Show a given param for all modules. You may change single params or set a param for all modules selected.
- `Traces`
  Accumulate triggered traces for all modules activated (one trace per module/channel). Data will be written to a `ROOT` file `trace.root`, and may afterwards be looked at via `HistPresent`.
- `Untrig Traces`
  Collect untriggered traces for all modules selected. Results are stored in file `untrigTrace.root`

- `Offsets`
  Start a "`ramping dacs`" task and adjust offsets. Untriggered traces should then have their baselines at 4 times the offset value.
- `MCA` (fig. 14)
  Start a `MCA` run. At end of accumulation histograms will be dumped on the ppc side, then converted to `ROOT` format and stored in file `mca.root`. To be looked at by `HistPresent`.
- `TauDisplay`
  Accumulate a number of triggered traces for selected module. Results will be displayed in a separate canvas.
- `Misc`
  Miscellaneous: Set/clear GFLT, set COINCWAIT
- `Save`
  Save dgf parameters to disk. Should be done on major changes to the dgf parameters.
- `Restore`
  Restore dgf parameters from disk
- `Copy`
  Copy certain parameters from one module/channel to others
- `Files`
  A list of files currently used
- `CPTM` (fig. 15)
  A panel to control the "Clock and Programmable Trigger" module (CPTM, Univ Cologne)

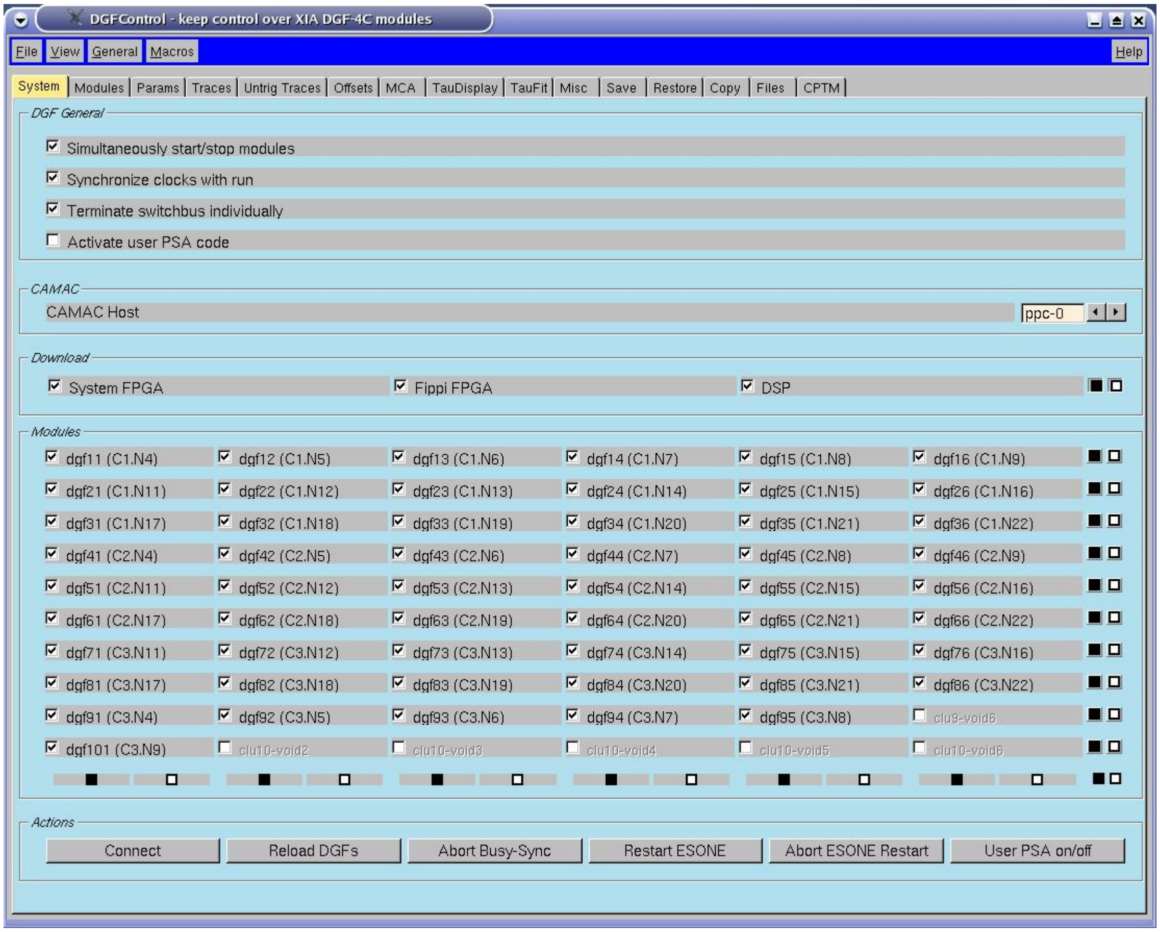


Figure 12: `DgfControl`: how to set up and control `XIA DGF-4C` modules

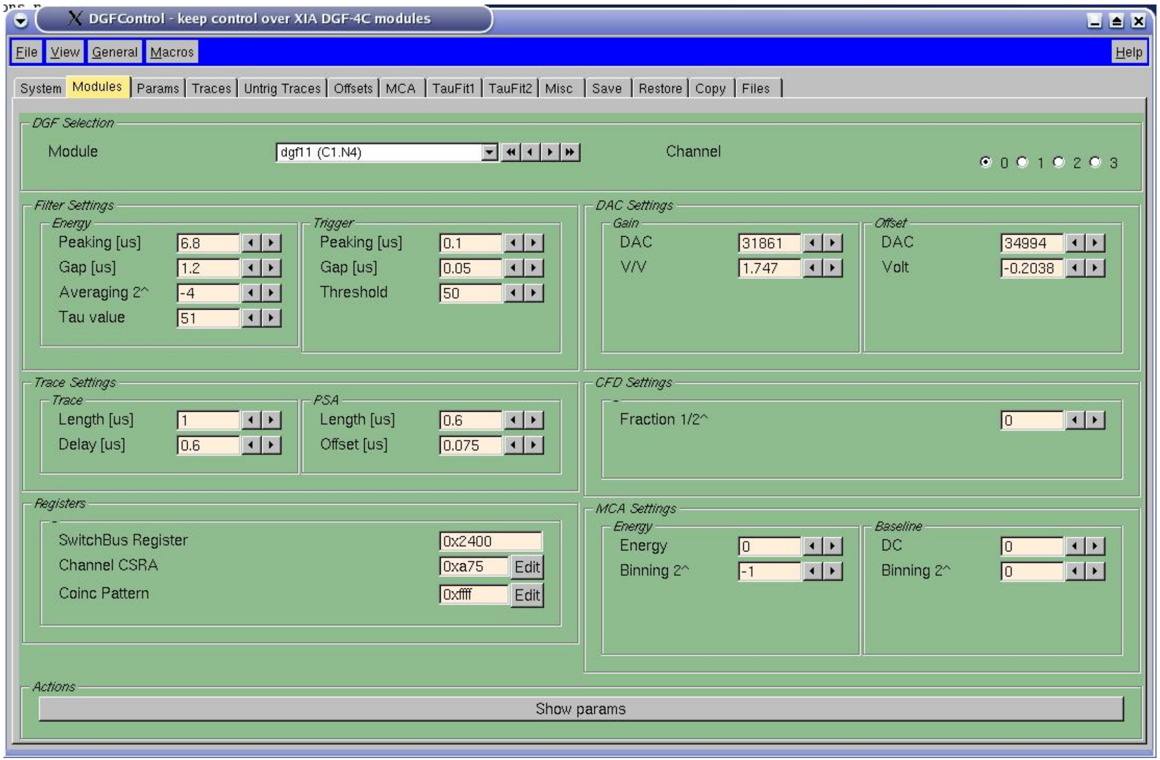


Figure 13: `DgfControl`: display parameter settings of `XIA DGF-4C` modules

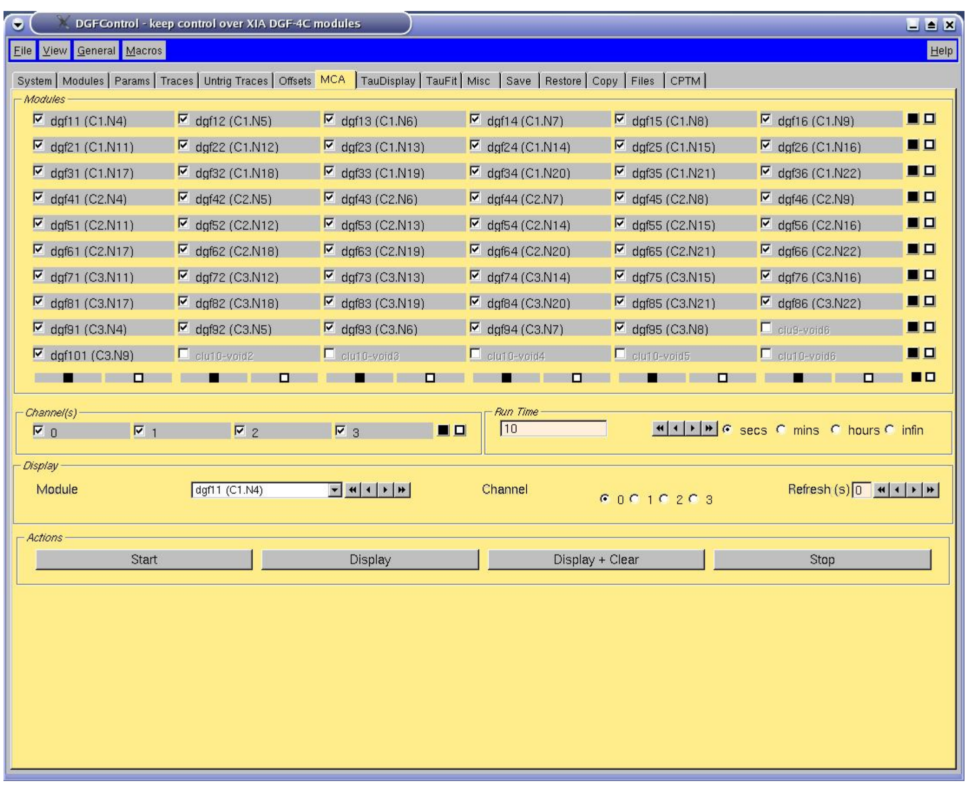


Figure 14: `DgfControl`: start a `MCA` accumation

Figure 15: `DgfControl`: how to control a `CPTM` module

3 How to perform an energy calibration

Oliver's program for energy calibration has now been modified to output data compatible with MAR$_a$BOU. So a conversion of the calibration data thru `olli2rudi` is no longer necessary.

To do an energy calibration (gamma or particle) call the `MacroBrowser`:

`MacroBrowser`

A menu will then pop up showing several `ROOT` macros. Choose `MBcal.C` from this list.

You'll get a form (fig. 16) to specify which type of calibration on which histograms you want to do:

- `Calibration`
  Choose `Co60` or `Eu152` for gammas, `TripleAlpha` for particles
- `Histo file / first histo`
  Click on the folder button and choose the `ROOT` file containing your calibration spectra. Choose histogram to start with.
- `Histo file / last histo`
  You have to select the **same** `ROOT` file as above. Choose the histogram to end with.
- `Calibration output file`
  where calibration data should be stored.
  This name should correspond to entries
  `TMrbAnalyze.CalibrationFile.DGF` (gamma) or
  `TMrbAnalyze.CalibrationFile.Caen` (particle)
  in your `.rootrc`. The extension has to be `.cal`.
- `Results file`
  where Oliver writes detailed calibration results
- `Precalibration file`
  To get an `Eu152` calibration you have to preform a `Co60` calibration step first. The name of the `Co60` calibration file has to be given here.
- `Verbose output`
- `Sigma for PeakFind` choose at least `5`
- `Relative percentage for PeakFind` `5`
- `Peaks to be fitted` `No`
- `Zero bins in front` `100`

Press `Execute` to start the calibration. Resulting calibration files will be read upon restart of your data acquisition (i.e. on next `Start` in `C_analyze`). For a description of the file format see 5.3.2

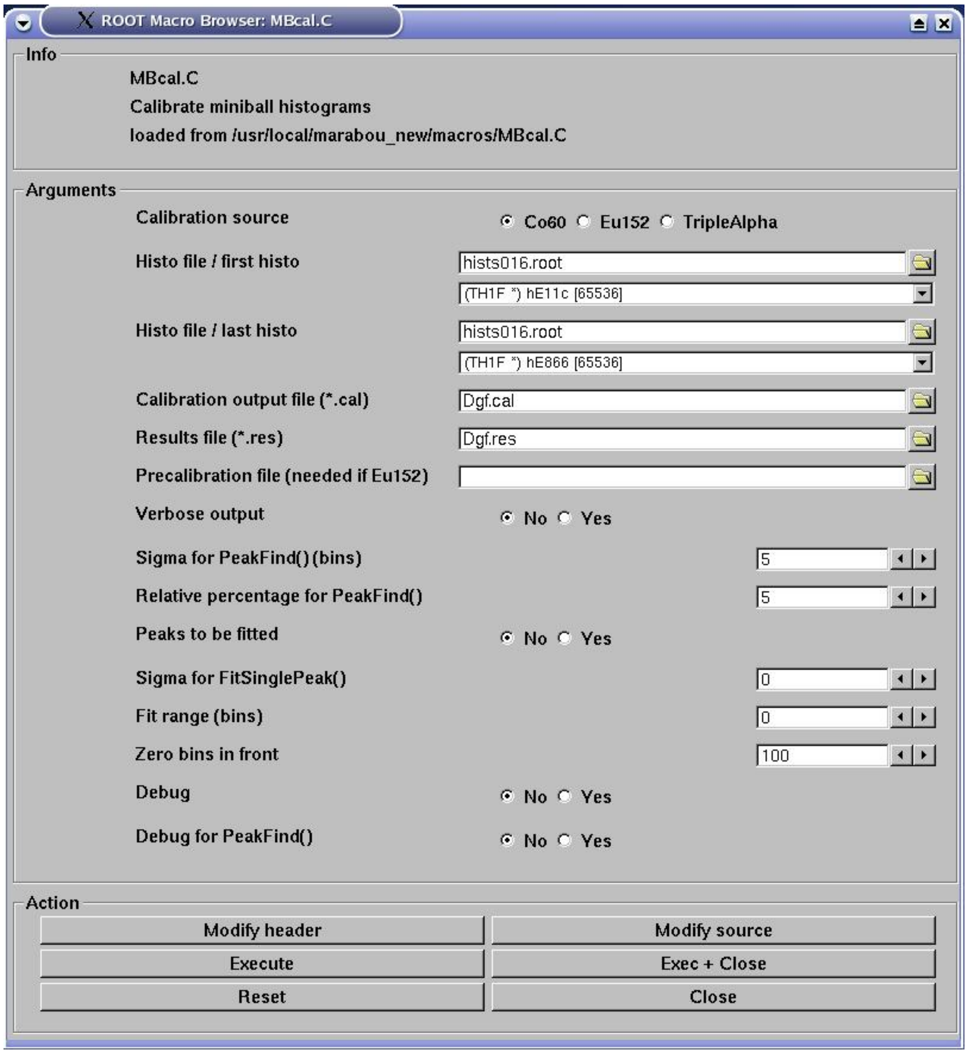


Figure 16: `MBcal.C`: how to do an energy calibration

4 How to do a Doppler correction

4.1 Doppler correction modes

To do a Doppler correction you have to create a file containing the correction coefficients for each histogram. A Doppler correction may be defined in three different ways.

- **Using a constant factor**
  You may have taken the Doppler shift from a fit to your histograms. The dcorr file then has one entry per histogram containing this factor.
- **Using a fixed geometry**
  If the particle is going in 0° forward direction the Doppler correction is given by the particle verlocity and the detection angle for cores and segments, respectively. Add one entry per histogram to the dcorr file containing this angle (degrees or radians).
- **Using a geometry depending on particle angle**
  You have to perform kinematic calculations to get velocity and angle for each particle independently. The dcorr file should then contain the detection angles with respect to 0° for each core and segment.

Add an entry

```
TMrbAnalyze.DCorrFile.DGF: <dcorr file>
```

to your `.rootrc`. The file extension has to be `.dcorr`. Doppler correction data will then be read from this file upon restart of your data acquisition (i.e. on next `Start` in `C_analyze`).

For a description of the file format see 5.3.3.

5 Appendix

5.1 Scripts

There are some scripts that should be run to monitor that everything works as expected. The purpose and how to start a certain script is explained below:

`scaler.sh`

This script displays the trigger scalers. See 1.4. It shows the rate with which some detectors or the DAQ are triggering.
Stop it by pressing Ctrl-C.

`dgfscaler.sh`

This script displays the internal DGF scalers (1.4).
Stop it by pressing Ctrl-C.

`ppac.sh` `[obsolete]`

Script to display the location of the beam in X and Y direction as measured with the PPAC.
No longer used, call `ppac.C` instead (1.5).

`start_rate_monitor.sh` `now WITHOUT leading "./"! [obsolete]`

Should be started once and should run all the time. It produces the files needed by the `plot_rate2.gp` script (see below). In case the rate plots are not updating even though the DAQ is running it might be that this script needs to be started again.
Script exits by itself.
Script is obsolete now – call `rateMon.C` instead (1.6).

`plot_rate2.gp` `now WITHOUT leading "./"! [obsolete]`

`Gnuplot` script that displays the 444 keV rate (bottom) and the beam dump rate (top) for 1, 5, 17, and 34 second averages.
Stop it by pressing Ctrl-C.
Script is obsolete now – call `rateMon.C` instead (1.6).

`monitor_rates.sh` *`threshold`* `now WITHOUT leading "./"! [obsolete]`

The keyboard bell rings and the screen flashes if the 17 second average 444 keV rate drops below the threshold given. In that case most likely the beam is gone and it should be checked if everything is still running.
Stop it by pressing Ctrl-C.
Script is obsolete now – call `rateMon.C` instead (1.6).

`CDThresPed2C <CDThres.ped >CDThres.C` `now WITHOUT leading "./"!`

Script used by `Config.C` during config step to convert pedestal file `CDThres.ped` to MARaBOOU commands in `CDThres.C`. Edit `CDThres.ped` according to your needs first.

`nigel2cluster` *`psFile cluFile`*

Script to convert Nigel's miniball config sheet from PostScript to ascii. `cluFile` is expected to have extension `.def`.

5.2 Files related to Config.C

5.2.1 Input files

To run a configuration step by executing `./Config.C` successfully several input files have to be present

- `in your working directory:`
  - `$HOME/.rootrc` and `.rootrc`
    contain `ROOT` resource definitions to control the config step
    define paths to other inputs like templates, macros, etc.
  - `cluster.def`, `cluster-void.def`, `other-dgfs.def`
    cluster definitions for active clusters, unsed clusters, and other dgf modules, resp.
    See 5.3.1 for file format.
  - `SetCppIfdefs.C` defines `#ifdef` settings for the cpp preprocessor:
    * Turn on event building (`online:off`)
    * Perform a window check for each event (`online:no`)
    * Use the CDE detector
    * Use a pattern unit
    * Used the beamdump detector
  - `BookHistograms.C` contains user's histogram definitions. Executed as part of `Config.C`.
    It is recommended to put any histogram defs in this file to increase readability.
  - `BookHistogramsOffline.C` (offline only)
    contains additional histogram defs for an `offline` session
  - `DefineVarsAndWdws.C` (offline only)
    defines windows and cuts for an `offline` session
  - `cluster.def` and `cluster-void.def`
    both define the cluster configuration to be used by `Config.C`. `cluster.def` contains active clusters as given by Nigel's configuration sheet, whereas `cluster-void.def` contains crate as well as station numbers for dgf modules currently unused (but present). Use script `nigel2cluster` to convert Nigel's PostScript file to `cluster.def`.
  - `other-dgfs.def`
    defines crate and station numbers for other dgfs such as time stamper and beamdump. Will also be read by `Config.C`.
- `in subdirectory config` (has to be part of resource `.rootrc:TMrbConfig.MacroPath`)
  - `UserMacro.C`
    contains user's code generation macros either as a replacement of or an extension to standard macros provided by MAR$_{a}$BOOU (change only if you are an expert).
  - several special templates used by `UserMacro.C` (don't touch either)
- `in subdirectory udef`
  - `BuildEvent.cxx/.h` how to build events from user's raw data

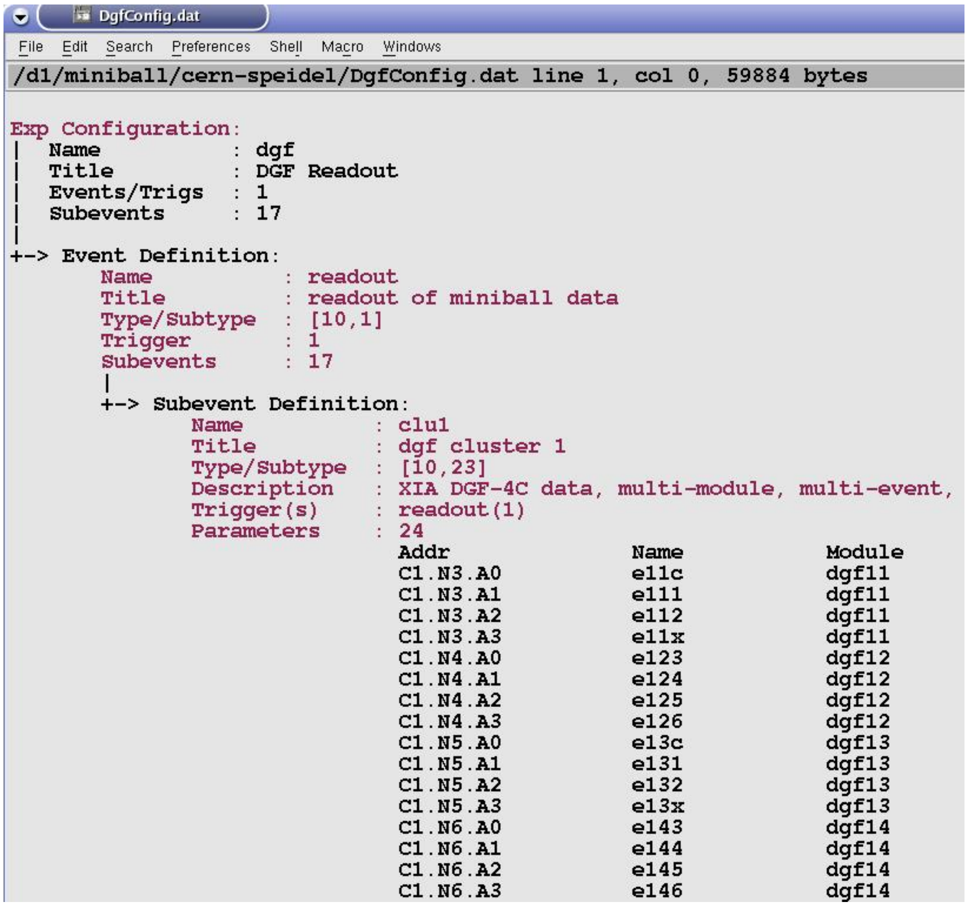
- `Analyze.cxx/.h` user's analysis event by event
- `TUsrHitEvent.cxx/.h` intermediate event structure during event building
- `Exp.h` final event structure after event building
- `HelpFunct.h` some helper functions
- `Winfo.h` window definitions

- `in the directory pointed to by resource .rootrc:TMrbConfig.TemplatePath`
  - template files to generate code for readout and analysis, respectively
  - templates files to generate files needed for MBS setup

5.2.2 Output files

Any output created by `Config.C` is written to the user's working directory. File names will be derived from the name of the config object in user's `Config.C`. If this object is named `dgf` for example, any file name created by `Config.C` will start with the prefix `Dgf...`. Any file starting with this prefix may be deleted without consequence - it can be re-generated by simply calling `./Config.C`.

- `.DgfConfig.rc`
  any names, counters, indices etc. created during the config step.
  Written using `ROOT`'s resource format.
  This file may be input by other service programs (such as `DGFControl`)
- `DgfCommonIndices.h`
  indices, serial numbers etc. to be used by both readout and analysis programs.
- `DgfReadout.c/.h`
  user's readout function loaded as part of `MBS` task `m_read_meb`.
- `DgfReadout.mk`
  Makefile to compile and link `MBS` task `m_read_meb` on ppc.
- `DgfAnalyze.cxx/.h`
  user's code generated automatically. Makes up main part of `M_analyze`.
- `DgfAnalyzeGlobals.h`
  any global definitions such as user's histograms, windows, variables.
- `DgfAnalyze.mk`
  Makefile to compile and link `M_analyze` for Linux.
- `.mbssetup`
  prototype file containing defs to set up `MBS`.
  Will be modified by `C_analyze`.
- `DgfConfig.dat`
  a printout of the configuration showing subevents, modules, and params (fig.17).



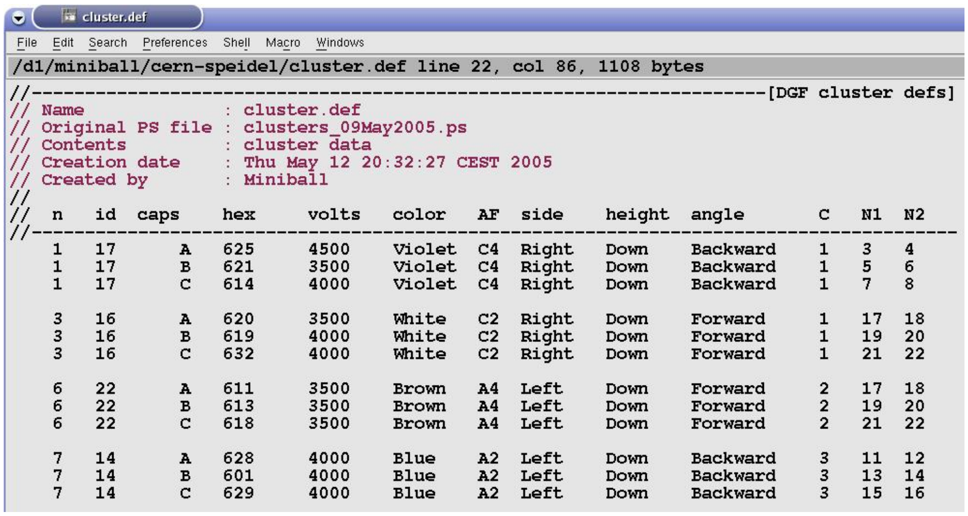
```
DgfConfig.dat
File  Edit  Search  Preferences  Shell  Macro  Windows
/d1/miniball/cern-speidel/DgfConfig.dat line 1, col 0, 59884 bytes

Exp Configuration:
|   Name           : dgf
|   Title          : DGF Readout
|   Events/Trigs   : 1
|   Subevents      : 17
|
+--> Event Definition:
        Name           : readout
        Title          : readout of miniball data
        Type/Subtype   : [10,1]
        Trigger        : 1
        Subevents      : 17
        |
        +--> Subevent Definition:
                Name           : clu1
                Title          : dgf cluster 1
                Type/Subtype   : [10,23]
                Description    : XIA DGF-4C data, multi-module, multi-event,
                Trigger(s)     : readout(1)
                Parameters     : 24
                                 Addr              Name            Module
                                 C1.N3.A0          e11c            dgf11
                                 C1.N3.A1          e111            dgf11
                                 C1.N3.A2          e112            dgf11
                                 C1.N3.A3          e11x            dgf11
                                 C1.N4.A0          e123            dgf12
                                 C1.N4.A1          e124            dgf12
                                 C1.N4.A2          e125            dgf12
                                 C1.N4.A3          e126            dgf12
                                 C1.N5.A0          e13c            dgf13
                                 C1.N5.A1          e131            dgf13
                                 C1.N5.A2          e132            dgf13
                                 C1.N5.A3          e13x            dgf13
                                 C1.N6.A0          e143            dgf14
                                 C1.N6.A1          e144            dgf14
                                 C1.N6.A2          e145            dgf14
                                 C1.N6.A3          e146            dgf14
```

Figure 17: `DgfConfig.dat`: printout of config data

5.3 Various file formats

5.3.1 Cluster definition files

The format of cluster definition files is adopted from Nigel Warr's Miniball Configuration sheet (fig.18).



```
//---------------------------------------------------------------------[DGF cluster defs]
// Name              : cluster.def
// Original PS file  : clusters_09May2005.ps
// Contents          : cluster data
// Creation date     : Thu May 12 20:32:27 CEST 2005
// Created by        : Miniball
//
//  n   id  caps    hex     volts   color   AF  side    height  angle       C   N1  N2
//-------------------------------------------------------------------------------------
    1   17     A    625     4500    Violet  C4  Right   Down    Backward    1   3   4
    1   17     B    621     3500    Violet  C4  Right   Down    Backward    1   5   6
    1   17     C    614     4000    Violet  C4  Right   Down    Backward    1   7   8

    3   16     A    620     3500    White   C2  Right   Down    Forward     1   17  18
    3   16     B    619     4000    White   C2  Right   Down    Forward     1   19  20
    3   16     C    632     4000    White   C2  Right   Down    Forward     1   21  22

    6   22     A    611     3500    Brown   A4  Left    Down    Forward     2   17  18
    6   22     B    613     3500    Brown   A4  Left    Down    Forward     2   19  20
    6   22     C    618     3500    Brown   A4  Left    Down    Forward     2   21  22

    7   14     A    628     4000    Blue    A2  Left    Down    Backward    3   11  12
    7   14     B    601     4000    Blue    A2  Left    Down    Backward    3   13  14
    7   14     C    629     4000    Blue    A2  Left    Down    Backward    3   15  16
```


Figure 18: `cluster.def`: define settings for DGF clusters

5.3.2 Energy calibration file

An energy calibration file generated by MBcal.C is formatted usind the ROOT resource format. It consists of a header followed by entries for each histogram.

Header	Calib.ROOTFile: <HistoFile>.root Calib.Source: Co60 or Eu152 or TripleAlpha Calib.NofHistograms: <N>
Entry (1 per histo)	Calib.<HistoName>.Xmin: <Value> Calib.<HistoName>.Xmax: <Value> Calib.<HistoName>.Gain: <Value> Calib.<HistoName>.Offset: <Value>

5.3.3 Doppler correction file

The dcorr file is formatted according to the ROOT resource format. It consists of a header followed by entries for each histogram / each parameter.

- Constant factor mode

Header	DCorr.Type: ConstFactor DCorr.NofHistograms: <N>
Entry (1 per histo)	DCorr.<HistoName>.Xmin: <Value> DCorr.<HistoName>.Xmax: <Value> DCorr.<HistoName>.Factor: <Value>

- Fixed angle mode

Header	DCorr.Type: FixedAngle DCorr.NofHistograms: <N> DCorr.AngleInDegrees: TRUE or FALSE DCorr.Beta: <Value>
Entry (1 per histo)	DCorr.<HistoName>.Xmin: <Value> DCorr.<HistoName>.Xmax: <Value> DCorr.<HistoName>.Angle: <Value>

- Particle-dependent angle mode

Header	DCorr.Type: VariableAngle DCorr.NofHistograms: <N> DCorr.AngleInDegrees: TRUE or FALSE
Entry (1 per histo)	DCorr.<HistoName>.Xmin: <Value> DCorr.<HistoName>.Xmax: <Value> DCorr.<HistoName>.Angle: <Value>